REMARKS

The invention relates to directed data flow "graphs" that represent an executable computer application. Each graph has vertices representing components and links between components indicating flows of data between such components (see: "The overall flow of information through such systems may be described in terms of a directed data flow graph, with vertices in the graph representing components (either data files or processes), and the links or "edges" in the graph indicating flows of data between components." (P. 1, ll. 7-10); "Again, the graphs will be composed of components (data files or processes) and flows (graph edges or links)." (P. 1, ll. 16-18)). As stated on p. 1, l. 20 over to p. 2, l. 5 of the application:

> *Graphs also can be used to invoke computations directly.* The "CO>OPERATING SYSTEM®" with Graphical Development Environment (GDE) from Ab Initio Software Corporation, Lexington, MA embodies such a system. Graphs made in accordance with this system provide methods for getting information into and out of individual processes represented by graph components, for moving information between the processes, and for defining a running order for the processes. This system includes algorithms that choose interprocess communication methods and algorithms that schedule process execution, and also provides for monitoring of the execution of the graph. (Emphasis added)

The invention formalizes the parameterization of data flow graphs to allow runtime parameters. Runtime parameters allow an application builder to defer the value of a parameter setting (*e.g.*, the key parameter of a sort function, file names, record formats, transform functions, *etc.*) to runtime (*i.e.*, the time an application is executed on a computer system). The values of runtime parameters may be supplied by the end user, or be derived from a combination of other runtime parameters or objects stored in an object repository, or be determined from a default setting.

Runtime parameters add flexibility to an application. Additional flexibility is achieved by using those parameters to compute metadata (data formats or types, and program logic or transforms) on demand. Types and transforms may be synthesized from other types and transforms, user-supplied parameter values, and stored objects (*e.g.*, from a repository). This makes it possible to build "generic" applications that work on input data of any type, or that

produce data through a series of transforms whose construction is controlled, directly or indirectly, through runtime parameter values.

One embodiment of the invention includes conditional components that permit changes to a graph structure (*and hence the application represented by the graph*) based on parameter values and computed metadata. Each component of a graph has a condition which controls whether or not that component will appear in the graph at runtime. The condition can be computed directly or indirectly through runtime parameters. Conditional components can be used to optimize or specialize graphs.

To make clear that the term "graph" is being used by Applicants in a consistent manner to mean a representation of an executable computer application having vertices representing components and links between components indicating flows of data between such components, Applicants have amended a number of the independent claims to essentially match the preamble of claim 1 as previously presented, and to reference execution of the application represented by the graph. In addition, claims 1, 14, and 27 have been amended to clarify that the invention modifies an executable computer application represented by a graph based on runtime parameters.

Claim Objections

Claim 27 was objected to. Claim 27 has been amended to obviate the objection.

Claim Rejections – 35 USC § 103

Claims 1, 3, 5, 9, 12-14, 16, 18, 22, 25-27, 29, 31, 35, and 38-39 were rejected under 35 U.S.C. 103(a) as being allegedly unpatentable over Sheard et al. (U.S. Patent No. 6,208,345, hereinafter "Sheard") in view of Poole et al. (U.S. Patent No. 6,006,242, hereinafter "Poole").

Claims 4, 8, 11, 17, 21, 24, 30, 34, and 37 were rejected under 35 U.S.C. 103(a) as being allegedly unpatentable over Sheard in view of Poole, and further view of Amado (U.S. Patent No. 5,701,400).

Applicants submit that the cited references, Sheard, Poole, and Amado, alone or in combination, fail to teach or suggest the invention as claimed.

Sheard teaches a visual interface for building a transport framework which facilitates the exchange of technology-dependent data between disparate applications. Sheard does not address the technology of graphs having runtime parameters or of modifying graphs. Specifically, Sheard does not involve a graph *representing an executable computer application and having vertices representing components and links between components indicating flows of data between such components*.

The Examiner persists in confusing the use of the depiction of a graph (meaning a "*Diagram showing varying quantities: a diagram used to indicate relationships between two or more variable quantities. The quantities are measured along two axes, usually at right angles. A graph may consist, for example, of a line joining points plotted between coordinates, a series of parallel bars or boxes, or a circle divided into wedges.*" See 2001 Encarta® World English Dictionary from Microsoft Corporation) with a data flow graph as defined by Applicants and used in the claims. The Examiner points to Sheard's "charting module" as generating basic system management queue charts. But such charts are <u>not</u> graphs representing an executable computer application and having vertices representing components and links between components indicating flows of data between such components. The present invention is not about generating charts – it involves representing an executable computer application in a particular way (which does not even need to be pictorially or graphically presented – a "graph" as used in the present application can be implemented as a text file that defines the component vertices and data flow links in text). Thus, while Sheard may teach use of runtime parameters in a particular context, there is no teaching or suggestion in Sheard to use or adapt runtime parameters to the technology of graphs such that an *executable computer application* represented by a graph is modified by such runtime parameters (*e.g.*, see claim 1, as herewith amended). Sheard cannot be fairly said to teach executing a graph using a final parameter value since Sheard does not discuss graphs (as defined by Applicants) at all.

Further, Sheard clearly does not teach or suggest *structurally modifying* a graph representing an executable computer application by *removing or replacing* conditional components (either data files or processes) at runtime (*e.g.*, see claims 9 and 12). This dynamic, runtime alteration of the very structure of an executable computer application represented by a graph is very much different from altering the display of a chart.

The portion of Sheard referenced by the Examiner as teaching modifying a graph in the manner claimed in fact simply shows two different views of the same application: FIG. 18 is a System Integration View selected by clicking on tab 520 (see FIG. 17) while FIG. 20 is a Business Integration View of the same thing, selected by clicking on tab 526 (Col. 20, ll. 3-30); the only difference between the views is that a statistical analysis icon is made visible in FIG. 20. FIG. 21 has nothing to do with determining conditional components, but instead shows a queue status chart developed from the data integration implementation shown in FIG. 18. The text cited in Col. 3, ll. 44-50 as teaching evaluating a condition mentions *nothing* of the sort; it simply says "The visual interface provides for runtime control and analysis of the business and technical aspects of an integrated information system deployment. Visual views onto the runtime system deployment provide consistent management and control for a variety of users having different data input/output, reporting, and interface requirements." A passing reference to "runtime control" does not enable practice of the present invention as claimed. Lastly, the text referenced in Sheard as teaching modifying a graph (col. 24, ll. 45-50) simply teaches that double-clicking on a component display displays charts of information trends – there is no structural change to a graph based on evaluation of a conditional component.

In contrast, claim 9 as amended requires modifying a graph by *removing* at least one conditional component, and all connected flows to such conditional component, from the graph. Claim 12, as amended, requires modifying a graph by *replacing* at least one conditional component with a flow before execution of the graph. These are structural changes to the application represented by the graph, neither of which are taught or suggested by Sheard. In both claims, the modified graph is then executed. Nothing in Sheard, alone or in combination, teaches or suggests such modification of executable graphs.

The Examiner admits that Sheard is insufficient alone to obviate claims 1, 3, 5, 9, 12-14, 16, 18, 22, 25-27, 29, 31, 35, 38, and 39 by having to combine Sheard with Poole. But Poole is directed at dynamic creation of electronic (*e.g.*, web pages) and printable documents, and teaches nothing about graphs representing an executable computer application or runtime modification of such graphs. While Poole mentions "graphics" and "graphical" a few times, it never discusses the concept of graphs as defined by the present application. Poole is simply directed ultimately to pictorial display of visual content, not substantive modification of an executable computer application at runtime. Thus, Poole is not even pertinent art, and does not add the elements admitted by the Examiner as being missing from Sheard.

The Examiner rejects dependent claims 4, 8, 11, 17, 21, 24, 30, 34, and 37 in view of the combination of Sheard, Poole, and Amado. Amado is directed to a system for applying artificial intelligence technology to data stored in databases and to generate diagnostics that are user definable interpretations of information in the database; there is no teaching or discussion of graphs in Amado in the sense defined by Applicants. Amado is cited only as teaching evaluation of an expression. However, for the reasons set forth above with respect to the failings of Sheard and Poole, the principal elements of these dependent claims are not obviated by Sheard and Poole, and thus adding Amado as teaching evaluation of an expression does not overcome the missing limitations in the principal references.

Please apply any charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: 9-14-04

Kenyon S. Jenckes
Reg. No. 41,873

PTO Customer No. 20985
Fish & Richardson P.C.
12390 El Camino Real
San Diego, California 92130
Telephone: (858) 678-5070
Facsimile: (858) 678-5099
10435636.doc